REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE 10/22/04	3. REPORT TYPE AND DATES COVERED Final 09/01/2001 – 08/31/2004

4. TITLE AND SUBTITLE Reducing the cognitive workload while operating in complex sensory environments.	5. FUNDING NUMBERS DAAD19-01-1-0754
6. AUTHOR(S) Leon N Cooper Predrag Neskovic	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Brown University Box 1843 Providence, RI 02912	8. PERFORMING ORGANIZATION REPORT NUMBER Final
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) U. S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709-2211	10. SPONSORING / MONITORING AGENCY REPORT NUMBER 42737.7-LS

11. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

12 a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	12 b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

A common characteristic of most artificial recognition systems in use today is that they are bottomup approaches, meaning that high-level modules do not influence processing of low-level modules. However, there are some problems and ambiguities at the level of sensory processing, and preprocessing of the signal, that cannot be resolved without taking into account cognitive level expectations. The major goal of our research within this project was to construct a functioning recognition system, based upon fundamental principles of human perception and cognition that exhibits the following properties: The ability to utilize contextual information during the recognition process. The ability to selectively focus attention on important regions of the input data. The ability to adapt preprocessing parameters to changes in the operating environment. The ability to engage in bi-directional information exchange with the user. The system that we constructed implements salient aspects of human perception and cognition such as saccadic eye movements, selective attention and cognitive level feedback during the recognition process, therefore having the potential to reduce the cognitive workload of the soldier when faced with the analysis of complex sensory environments. In this report we provide a brief description of the models and algorithms that we developed within the scope of this project and we present the performance of those models when tested on real-world datasets. We conclude the report with the summary of the most important results.

14. SUBJECT TERMS recognition system, signal processing, perception	15. NUMBER OF PAGES 42
	16. PRICE CODE

17. SECURITY CLASSIFICATION OR REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION ON THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev.2-89)
Prescribed by ANSI Std. 239-18
298-102

Contents

List of Figures

List of Tables

1 Statement of the Problem Studied

A common characteristic of most artificial recognition systems in use today is that they are bottom-up approaches, meaning that high-level modules do not influence processing of low-level modules. However, there are some problems and ambiguities at the level of sensory processing, and preprocessing of the signal, that cannot be resolved without taking into account cognitive level expectations. The question, both theoretical and practical, is then how to utilize contextual information on various levels of processing of information. In this report we provide a biologically inspired solution to this problem and show that it can be applied to several different domains.

The major goal of our research within this project was to construct a functioning recognition system, based upon fundamental principles of human perception and cognition, that exhibits the following properties:

a) The ability to utilize contextual information during the recognition process

b) The ability to selectively focus attention on important regions of the input data

c) The ability to adapt preprocessing parameters to changes in the operating environment, and

d) The ability to engage in bi-directional information exchange with the user.

The system that we constructed implements salient aspects of human perception and cognition such as saccadic eye movements, selective attention and cognitive level feedback during the recognition process, therefore having the potential to reduce the cognitive workload of the soldier when faced with the analysis of complex sensory environments.

In the next several sections we provide a brief description of the models and algorithms that we developed within the scope of this project and we present the performance of those models when tested on real-world datasets. We conclude the report with the summary of the most important results.

2 Detection of Moving and Still Vehicles from Real-Time Videos

2.1 Introduction

Identification of vehicles from video streams is a challenging problem that incorporates several important aspects of vision including: translation and scale invariant recognition, robustness to noise and occlusions and ability to cope with significant variations in lighting conditions. In addition, the requirement that the system work in real-time often precludes the use of more sophisticated but computationally involved techniques.

The problem of vehicle identification from video streams has been widely addressed in computer vision literature (Koller et al., 1994; Lipton et al., 1998a; Koller et al., 1993b). Very often, an underlying assumption is that the vehicles are moving and motion information is used to segment the image into moving regions and a static background. Based on its overall size and shape, a region can then sometimes be recognized as a vehicle even without a detailed description. Furthermore, motion information can reduce the computational complexity since only the regions that contain motion have to be analyzed. However, in many situations, motion information is not available or is insufficient, and other ways of dealing with computational complexity and segmentation problems have to be used. In contrast to motion-based models for vehicle detection (Koller et al., 1994; Lipton et al., 1998a), our approach does not rely on motion information, and the system can detect both still and moving cars in real-time.

Several approaches use edge information in order to detect vehicles either from still images or video streams. Betke et al. (M. Betke and Davis, 1997; M. Betke and Davis, 2000) make use of motion and edge information to hypothesize the vehicle locations. However, a vehicle is not represented as a collection of edges. Instead, the edges are used only in order to capture the boundary of a vehicle. Once a region is hypothesized as vehicle, the recognition is performed by matching a template with the potential object marked by the outer horizontal and vertical edges. Edge information is also contained in Haar wavelets that were used as features supplied to Support Vector Machine in (Papageorgiou and Poggio, 1999; Z. Sun and Miller, 2002). Similarly, Goerick et al. (C. Goerick and Werner, 1996) use Local Orientation Coding (LOC) to extract edge information. The histogram of LOC within the area of interest was then fed to a Neural Network (NN) for classification.

Other researchers have also used feature-based (or parts-based) approaches to detect vehicles from images. Wavelets were chosen as features in (Papageorgiou and Poggio, 1999; Z. Sun and Miller, 2002; Schneiderman and Kanade, 2000), rectangle features (which are similar to Haar wavelets) were used in (Viola and Jones, 2001) while the authors in (Webber et al., 2000; Agarwal and Roth, 2002) use the interest operator for automatically selecting features. In (Webber et al., 2000), a generative probabilistic model is learned over the selected features and, due to a computational complexity, the method they use relies on a very small number of fixed parts.

Biologically inspired approaches have been much less successful than computer-vision or statistical approaches when applied to real-word problems. Biologically-based recognition systems have been proposed for various applications such as face recognition, handwriting recognition and vehicle detection (S-W. Lee and (Eds.), 2000). An approach to object recognition that is based on human saccadic behavior is proposed in (Keller et al., 1999). While this model does capture properties of saccadic behavior, it represents an object as a fixed sequence of fixations. In contrast to this approach, our system does not make such an assumption and detects a car regardless of the order in which saccades were perform.

Numerous experiments and computational theories (Logothetis and Sheinberg, 1996; E. Wachsmut and Perrett, 1994; Biederman, 1987) advocate the representation of objects in terms of parts or features and relations among these features. In particular, edges have been proposed as the most basic features for object representation. Compared to pixels, edges are much more stable, less susceptible to noise and changes in lighting conditions, which make them good candidates for image analysis and representation. Since the classical experiments of Hubel and Wiesel (Hubel and Wiesel, 1968), in which they discovered neurons in visual cortex that were selective to edges of various orientations and sizes, edges became important ingredients in understanding the biological processing of information. From an information-theoretic perspective, edges can be viewed as independent components of natural scenes (Bell and Sejnowski, 1997) and thus they provide compact and sparse representation of visual images. This suggests that edges are important blocks in representing not only visual scenes, but also objects within a scene, and that edge-based representation is both biologically and mathematically reasonable. However, the extraction of edges remains a fairly difficult task for several reasons.

The first problem is related to the fact that given the local information in the region of an image it is very difficult to say whether the region contains an edge or not. Human perception of an edge is so much influenced by context that we sometimes ”see” edges where there are none. Computer algorithms usually don’t rely on contextual information and utilize only bottom-up information. Most edge operators are very dependent on two parameters: the scale and the threshold. The problem is that none of these two parameters can be set a-priori or globally. The question is how to provide context, or in other words, how to dynamically estimate the scale and threshold value.

The other problem is of a computational nature. Unlike the visual cortex that extracts edges of all orientations and sizes at all locations at every moment in parallel, the computations in current computers are performed serially. Therefore, unless a specialized hardware is used, edge extraction is a computationally very expensive task in real-time applications.

In constructing an artificial recognition system for real-time processing of video streams we draw inspiration from the way the human visual system analyzes complex scenes. The properties of the human visual system that we utilize in our system are: representation of objects in terms of parts, selective attention, saccadic eye movements and hierarchical processing of visual information.

Due to the structure of the eyes, the human visual system does not process the whole visual input with the same resolution. The region of the scene that is perceived with the highest quality is the one that projects to the fovea, an area of the retina corresponding to only about the central 2 degrees of the viewed scene. The regions that are further away from the fixation point are perceived with progressively lower resolutions. The visual system overcomes this limitation by making rapid eye movements, called *saccades*. Human recognition is therefore an active process of probing and analyzing different locations of the scene at different times and integrating information from different regions.

In our model, an object (a car) is represented as a collection of horizontal and vertical edges arranged at specific spatial locations with respect to each other. Within a single fixation, the locations of features are always measured with respect to the fixation point - the central edge. During the recognition process, the system efficiently searches the space of possible segmentations by investigating the local regions of the image in a way similar to human eye movements.

2.2 Segmentation, Grouping and Selection of the Threshold Value

One of the most important problems related to the detection of cars from video streams is the segmentation problem. Given an image, it is not known where a car is or what its size is. Therefore, all the methods that assume a fixed size input vector during the recognition, (e.g. NNs), are not very well suited for this problem.

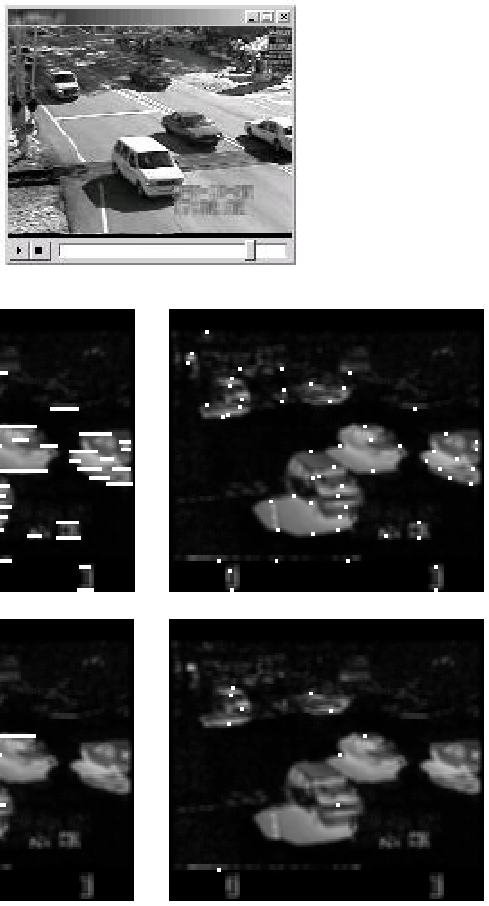


Figure 1: Top: The original image. Second row (left): Some of the horizontal edges extracted using the low threshold value. Second row (right): Estimated centers of the extracted edges (from the left image). Third row (left): Some of the horizontal edges extracted using the high threshold value - the strong edges. Third row (right): Estimated centers of the extracted edges (from the left image). All of the edges in the second and third row are displayed over the difference image for the clarity of presentation although the edges were extracted from the original (top) image.

In order to detect a vehicle regardless of its location, the detection system has to be convolved over the whole image, and in order to detect a vehicle at different scales the original image has to be

rescaled and the convolution procedure repeated (Papageorgiou and Poggio, 1999; Schneiderman and Kanade, 2000). Since the methods that rely on the exhaustive search are not very efficient they are mostly applied to detection of vehicles from static images.
In case of NNs, the difficulty arises both during the training phase and during the testing phase. One of the important questions related to the preparation of training samples is: how much of the vehicle should be present in the training window? In order for the network to be able to recognize occluded vehicles it would help if it is trained on parts of vehicles. However, not only that it is difficult to define the "minimal size" of the part of the vehicle that should be present in the window but it is much more difficult to train the network on parts of vehicles. During the recognition phase, the network will give multiple detections corresponding to a single vehicle in the region around the vehicle. This problem is not unique to NNs but to all the methods that assume fixed size input window ((Papageorgiou and Poggio, 1999; Z. Sun and Miller, 2002; Schneiderman and Kanade, 2000; Viola and Jones, 2001; Agarwal and Roth, 2002)). Possible remedies are suggested in (Viola and Jones, 2001; Agarwal and Roth, 2002).
One solution to the segmentation problem is to represent a vehicle in terms of its parts or features and to extract them from the whole image (Webber et al., 2000) as opposed to the window of a specified size as in (Papageorgiou and Poggio, 1999; Z. Sun and Miller, 2002; Schneiderman and Kanade, 2000; Viola and Jones, 2001; Agarwal and Roth, 2002). However, then the problem is how to group or select only a portion of the features from the entire image and this can be a serious computational issue, especially if the number of features per object is large (Webber et al., 2000).
Related to the grouping problem is the selection of the threshold value for edge extraction. Choosing the low threshold value allows extraction of most correct edges but also a large number of spurious edges are extracted which in turn creates a combinatorial explosion when one tries to group the edges. On the other hand, setting the threshold value too high alleviates the binding problem but results in too few edges extracted for reliable recognition. We will illustrate this point in the following example. Let us assume that the object that we want to recognize is a car and that it is represented as a collection of edges of different sizes and orientation arranged at specific locations with respect to each other. In order to recognize a car one has to 1) reliably extract edges and 2) group some of the extracted edges such that they represent a car.
We used the top image (the original) as shown in Figure 1 as a starting point and only the horizontal edges were extracted by convolving the image with a Prewitt operator mask. The points above a specific threshold value were taken as edge points and then edge sizes and the locations of their centers were estimated. The edges and their centers, shown in the second and third row in Figure 1, were extracted using two different threshold values. The threshold value used in the second row is much smaller than the one in the third row and therefore more edges were extracted. All the edges are displayed over a difference image (the original image minus the image without vehicles) for the purpose of clarity, although the edges themselves were extracted not from the difference image but

from the original image.
As it can be seen from the left image in the third row, most of the extracted edges belong to cars but there are too few edges and it is not possible to reliably detect cars based on those edges. In the left image of the second row, on the other hand, most of the car edges were correctly extracted, but the problem is that in addition to the correct edges there are a lot of non-car edges such as the edges at the boundary of the shadow or the edges from the road. There are too many edges and it is very difficult to figure out which edge goes with which edge - how to group or bind the right edges. This problem becomes much worse when edges of other directions are also extracted and it becomes a computational issue, especially for real-time applications. In addition to this problem, the edge extraction over the whole image, for every single frame, is computationally expensive without dedicated hardware.
One possible solution to these problems is to design a system that emulates properties of human vision such as rapid eye movements, selective attention and hierarchical processing. As opposed to detecting the whole object in the scene, in most cases, human recognition starts with identification of one or more prominent features, and only when there is a need for extracting finer details, additional resources are employed. In application to car detection this suggests the following scenario for the recognition process: 1) Start with extraction of strong edges (e.g. the third row in Figure 1) and therefore use high threshold value. 2) Select one or more of them and make a hypothesis as to what they represent. This is a hierarchical approach towards segmentation starting with edges that are less likely to be incorrect. 3) If more information is needed to confirm the hypothesis, search for additional edges but only of specific orientations and sizes and within the local regions of the image where they are expected to be found. This means that the threshold value is dynamically adjusted, and lowered, only within specific regions where their presence is very likely. 4) If the hypothesis is not supported, choose another edge or a group of edges until all of the prominent edges are examined. This is the procedure that we followed in constructing our recognition system (Neskovic and Cooper, 2002; Neskovic and Cooper, 2003; Rajapakse and (Eds.), 2004)and in the following sections we will describe it in more detail.

2.3 The Model

In our model, an object is represented as a collection of features of specific classes arranged at specific spatial locations with respect to the fixation point. In the current application, car detection from video images, the features are the edges of different orientations and sizes. In contrast to other models that also view an object as a collection of local features (Webber et al., 2000; Wiskott et al., 1997), the positions of the features in our model are always measured with respect to the location of one feature that we call the central feature (the central edge). During the recognition process the central edge becomes the edge on which the system fixates. We further assume that saccades cannot be made on any point in the image but only on edges, more specifically on their centers.

Once a saccade is made on an edge the question is then how do we know which edge of a car it represents? Obviously, from the strength of the edge itself it is not possible to answer that question since edge strength is obtained using only local and bottom-up information. What is needed is a presence of other (car) edges, to put an edge in context, and we want to know how to measure their influence.

Due to the fact that cars come in different shapes and sizes, the location (and size) of any edge is not fixed with respect to the location of a chosen central edge. So, for a given edge (the central edge), the location of any other edge is not at any specific location but within a region. These uncertainties in edge locations can be calculated from car statistics and can be expressed through conditional probabilities.

Consider two edges of a car, e.g. the bottom edge below the grill (denote it as e_i) and the top edge above the windshield (denote it as e_j). Given the location of the bottom edge, $\vec{r_i}$, the position of the top edge, $\vec{r_j}$, varies due to variations in car sizes. The collection of all the possible locations of the top edge forms a region (R_j) where the top edge can be found given the location of the bottom edge. If we assume that all the locations of the top edge within the region are equally likely then the probability of finding the top edge at any place within the region R_j is inversely proportional to the size of the region $p(e_j|e_i, \vec{r}_{ij}) = cons * 1/S(R_j)$, where $S(R_j)$ denotes the size of the region R_j and $\vec{r}_{ij}$ is the location of the center of the j^{th} edge with respect to the location of the i^{th} edge. The edge e_i provides context for the existence of the edge e_j in the region R_j and its influence is expressed through the conditional probability $p(j|i, \vec{r}_{ij})$. Similarly, the edge e_j provides context for edge e_i with strength $p(e_i|e_j, \vec{r}_{ji})$, where $\vec{r}_{ji} = -\vec{r}_{ij}$. It can be easily shown that $p(e_j|e_i, \vec{r_{ij}}) = p(e_i|e_j, \vec{r_{ji}})$. In the rest of the paper we will use abreviated notation $p(e_j|e_i, \vec{r_{ij}}) = p(e_j|e_i)$ assuming the spatial dependence.

Let us consider the i^{th} edge of a car and assume that it is detected with probability d_i at some location in the image. In order to increase the confidence about the identity of this edge, it would help if every other edge of the vehicle is detected at its expected locations and with high confidence. The support from other edges can be incorporated in a number of ways and one of the simplest is to average their contributions. We define a "context" for the i^{th} edge to be the sum $\sum_{j=1, j\neq i}^{N} p(e_i|e_j)d^j$ divided by the number of contributing edges, where N is equal to the number of edges representing a car.

The collection of conditional probabilities, associated with a given central edge thus constitutes a model of a car from the point of view of the particular central edge. We can think of the collection of local regions where the edges are expected to be found as a template that represents a car from the point of view of the given central edge. Therefore, a car is represented with as many templates as there are edges that can be used as fixation points. The conditional probabilities between the edge pairs are mapped into the weights of a neural network and the task of the network will be to combine the information coming from the edge detectors (bottom-up) with expectations for edge locations (top-down).

2.4 The Architecture of the Network

In this section, we describe the architecture of the network that represents a car. We will assume that every edge (from an image) can be used as a fixation point during the recognition process and with each (car) edge we will associate one unit, an object-unit, that will represent an object from the point of view of that edge. We will now focus on one such object-unit and the group of units from which it receives inputs as illustrated in Figure 2. At the bottom of the hierarchy are

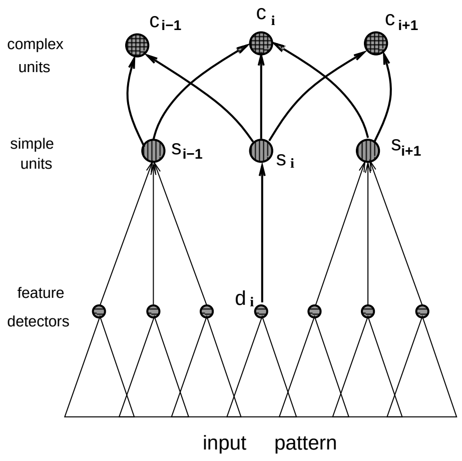


Figure 2: The network architecture. An i^{th} object unit represents a car from the perspective of an i^{th} edge of a car. There are as many object units as there are possible views of a car, which is equal to the number of edges that represent a car. Similarly, there are as many complex units as there are edges in a car and each complex unit combines bottom-up information and contextual information. The i^{th} complex unit combines bottom-up information coming from the edge detector selective to the i^{th} edge of a car with information coming from all other simple units that represent other edges of the car. In contrast, any other complex unit incorporates contextual information coming only from the central edge. Simple units, on the other hand, receive only bottom-up information and find the strongest edges, to which they are selective, within their receptive field. The sizes of the receptive fields of the simple units get progressively larger, further away they are from the location of the central edge (fixation point), thus allowing variations in edge locations (car shapes and sizes).

the edge detectors whose receptive fields completely cover the input image. An output of an edge detector is the probability, d, of detecting the edge to which it is selective, e.g. an edge of specific orientation and size. The outputs of the edge detectors are supplied to the next layer of units - the *simple units*. Among the simple units, we distinguish a central unit, the one that is positioned above the fixation point, and the surrounding units. The size of the receptive field of the central

unit is the smallest, when compared to other simple units and the sizes of the receptive fields of the simple units increase with their distance from the central unit. The sizes of the receptive fields of the simple units are designed in such a way as to accommodate the uncertainties associated with locations of the edges with respect to the fixation point.
The output of a simple unit, given the location of the i^{th} central unit, is given as

$$s_{ij} = max_{\vec{r} \in R_j}(d_j(\vec{r})), \tag{1}$$

where $\vec{r}$ is the location of the edge detector (selective to the j^{th} edge) with respect to the location of the central edge (that represents the i^{th} edge of a car) and R_j is the receptive field of the j^{th} simple unit.
Therefore, a simple unit selects the strongest edge within its receptive field and outputs the probability that this edge represents the j^{th} edge of a car. The next layer of the units, called the *complex units*, incorporates contextual information. The complex unit that receives input from the central simple unit outputs the probability that the region R_i (or the edge it contains) now represents part of the object

$$c_{ii} = d_i \frac{1}{N-1} \sum_{j=1, j \neq i}^{N} p(e_i|e_j) s_{ij}, \tag{2}$$

where N represents the number of edges in the object. This means that the detection of the central edge is now viewed within the *context* of all the other edges of the object. Similarly, the j^{th} complex unit that receives input from the j^{th} simple unit views the j^{th} edge within the context of the central edge

$$c_{ij} = d_j p(e_j|e_i) d_i. \tag{3}$$

According to our model, each local region (an edge) can represent an object with different confidence. The probability that the collection of all the regions that contain object edges represents the object from the point of view of the i^{th} edge is captured by the *object unit*

$$o_i(object|fixation\ point\ i) = \frac{1}{N} \sum_{k=1}^{N} c_{ik}, \tag{4}$$

where the index k goes through all the complex units, the central (i) and surrounding (j) units, of a given view. It is clear that there are as many object units as there are possible views of the object, which in our case, is equivalent to the number of edges in the object.

2.5 Implementation

Ideally, the system would contain an array of feature detectors that completely cover the input image and process information in parallel. Similarly, the system would benefit from a large number of feature classes, since they would provide richer and more detailed description of objects. However, in order to make a system run in real time on a regular computer and without dedicated preprocessing hardware, we had to make several approximations.

2.6 Feature Selection

Due to the shape and structure of the vehicles, it seems natural to choose edges of various sizes and orientations and corners as representative features. However, through our experiments, we learned that extraction of edges at all the angles is prohibitively expensive if the system is to run on real time. The problem with corners is that they are much more susceptible to noise and therefore much more unstable compared to edges. Therefore, in our current implementation, we represent a car as a collection of only horizontal and vertical edges. Since an edge is an extended spatial object, we choose to specify its location in terms of the location of its central point. In this way, a car is modeled as a collection of points, arranged in 3D space, where each point represents an edge of specific size and orientation. Using the statistics for the car sizes and their edges, one can easily calculate the mean size μ_j and the variance ν_j for each edge. Given the location of the fixation point and knowing the variations in size for every edge, it is then straightforward to propagate these uncertainties and calculate the regions where the centers of the edges should be. In order to map this 3D configuration of regions into a 2D image plane we use perspective transformation Eqs. (5) - (7). In this way, for a given location of the fixation point within an image, we associate a group of 2D regions as being allowable locations for the edge centers. Each such region represents a receptive field of one simple unit of the network.

2.7 Perspective Transformations

The perspective transformation equations that we use are described in more detail in (Gonzales and Woods, 1993). Here we briefly review the main equations. Let us denote with $(x_p^{'}, z_p^{'})$ and with $\vec{v} = (x, y, z)$ the coordinates of a point in an image plane and in the real-world respectively. The location of the camera (the gimbal center) is at the point $\vec{v_o} = (x_o, y_o, z_o)$. The vector $\vec{l} = (l_1, l_2 + f, l_3)$ denotes the constant offset between gimbal center and image plane center where f is a constant offset along the optical axis. The world coordinates are transformed into image coordinates using the direct perspective transformation equations:

$$x_p^{'} = f \frac{(x - x_o)cos\theta + (y - y_o)sin\theta - l_1}{-(x - x_o)cos\psi sin\theta + (y - y_o)cos\psi cos\theta + (z - z_o)sin\psi - l_2}, \tag{5}$$

$$z_p^{'} = f \frac{x - x_o)sin\psi sin\theta - (y - y_o)sin\psi cos\theta + (z - z_o)cos\psi - l_3}{-(x - x_o)cos\psi sin\theta + (y - y_o)cos\psi cos\theta + (z - z_o)sin\psi - l_2}. \tag{6}$$

The image coordinates are transformed into world coordinates using the inverse perspective transformation equations:

$$\vec{v} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_o + l_1 cos\theta - l_2 cos\psi sin\theta + l_3 sin\psi sin\theta \\ y_o + l_1 sin\theta + l_2 cos\psi cos\theta - l_3 sin\psi cos\theta \\ z_o + l_2 sin\psi + l_3 cos\psi \end{bmatrix}$$

$$+\lambda \left[\begin{array}{c} x_p^{'}cos\theta - fcos\psi sin\theta + z_p^{'}sin\psi sin\theta \\ x_p^{'}sin\theta + fcos\psi cos\theta - z_p^{'}sin\psi cos\theta \\ fsin\psi + z_p^{'}cos\psi \end{array} \right], \tag{7}$$

where the λ is a free non-negative parameter. Knowing the real coordinates of a point (x, y, z) it is easy to calculate the image coordinates $(x_p^{'}, z_p^{'})$ using Eqs. (5)-(6). However, in order to go from the image coordinates to the real coordinates one has to provide more information. In most cases it is assumed that the point is on the groud and therefore $z = 0$. Solving Eq. (7) for λ (by setting the third component to zero) and substituting it back in (7) one easily arrives at the values for image coordinates.

2.8 Feature Detectors

Another approximation is related to the construction and use of edge detectors. Instead of having an array of edge detectors for detecting the horizontal and vertical edges of all the sizes, the system extracts only the prominent edges (with activations above the predefined threshold) and estimates their sizes. Figure 3 illustrates some of the extracted edges and their estimated sizes. In our current implementation, the edges were extracted from the difference image that is obtained as a difference between the original (gray-scale) image and the background image that contains no vehicles.

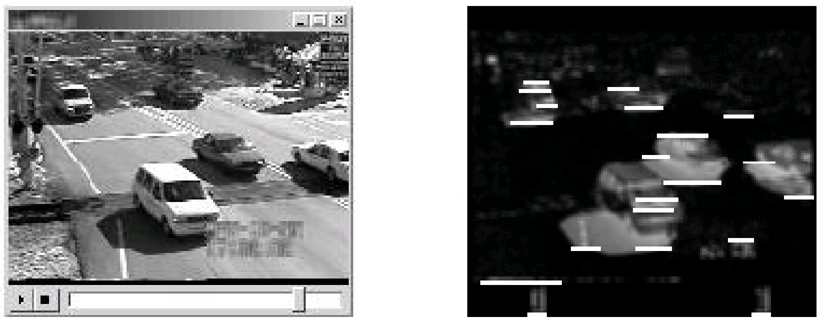


Figure 3: Original image (left) and processed image (right) that illustrates some of the prominent horizontal edges. The extracted edges (right image) are projected on the difference image that is obtained by subtracting the background image (that contains no vehicles) from the original image. This is done only for the illustrative purposes - to make the extracted edges more visible.

The value of the pixel (i, j) of the background image at time $t + 1$ is calculated using the updating rule

$$B_{t+1}(i, j) = B_t(i, j) + \alpha \cdot D_t(i, j) \cdot O(i, j), \tag{8}$$

where α is an updating constant, $D_t(i, j)$ is the difference between the pixel values at times $t + 1$ and t and $O(i, j)$ is 0 if the pixel (i, j) belongs to an object that has been identified and 1 if it is

part of the background. Therefore, the current image is used for updating the background image after the object identification is performed on the current image.
Although the structure of each edge is fairly simple, there are still several characteristics that can be used as possible descriptors. For example, an edge can be characterized by its size, or its intensity or with both characteristics and the question is which of those characteristics is the most appropriate. As a measure of the importance of each of those characteristics, we use the mean values and variances of the edges that belong to vehicles and edges that do not belong to vehicle. In order to collect those values we designed a new module that allows the user to interactively label all the edges extracted by the system (both vehicle and non vehicle edges) and store their values for each frame while the program is running. More detailed description of the interactive module will be given in the following section. As can be expected, the values of edge size vary much less compared to values in edge intensities due to the large variations of light intensities and variations in vehicle colors. Therefore, in our current implementation, we characterize edge only in terms of their sizes.
Each edge detector is selective to only an edge of a specific orientation, but can detect edges of various sizes around the preferred size. Since the distribution of sizes for any given car edge is fairly uniform, we use a Gaussian distribution to model the probability of an edge having a specific size. Therefore, an edge detector for an edge of horizontal/vertical orientation is specified with two parameters: the mean length of an edge and its variance. The input to the edge detector (of a given orientation) is an edge of specific size l and the output is the probability that measures how well this edge matches the expected edge size, $d = 1/(\nu\sqrt{2}\pi) * exp(-(\mu - l)^2/2\nu^2))$.

2.9 Recognition Process and Results

During the recognition process, the system explores the input image in a way similar to human saccadic eye movements, probing and analyzing different locations of the input at different times. Since the extraction of edges of all possible orientations and sizes at all image locations is computationally unfeasible, we use a two-stage approach towards edge extraction and in the first stage only the strong edges are extracted. This means that the edges with activations above some predefined threshold value are extracted and then the sizes of these edges and their centers are estimated using only local information. This is a purely bottom-up stage. As a result, a number of edges are not extracted because they are not strong enough and the sizes of many edges are estimated incorrectly, as illustrated in Figure 1 the third row. These edges are therefore used only as a starting point for the recognition process. The recognition system saccades on one of the prominent edges and this edge, together with neighboring edges, activates all object units. In Figure 4 (a) we presented only the central edge without the neighboring edges although both the central edge and the strong neighboring edges, if present and at the correct locations, activate an object unit. Each object unit with activation above some predefined threshold value provides context for the next stage of edge extraction in the sense that the presence of strong edges provides context and support for the presence of fine edges (the edges with low strength), Figure 4 (c). Given a location of the central edge, an image is partitioned into local regions where the system expects to find edges of specific orientations and sizes, Figure 4 (d). Each of these regions is then convolved with an edge operator mask and the maximal value of the convolution within a region is taken as the representative of the edge strength within that region. The object unit with the highest activation selects some of the surrounding edges (around the central edge) as representative of a car, and the central edge is given the identity as being a specific edge of the car (e.g. the bottom horizontal edge). Activations of all the edges covered by the receptive fields of the complex units that belong to the same object unit are then suppressed and the system chooses another prominent edge as the next fixation point. This procedure is repeated until all of the strong edges within an image are examined.

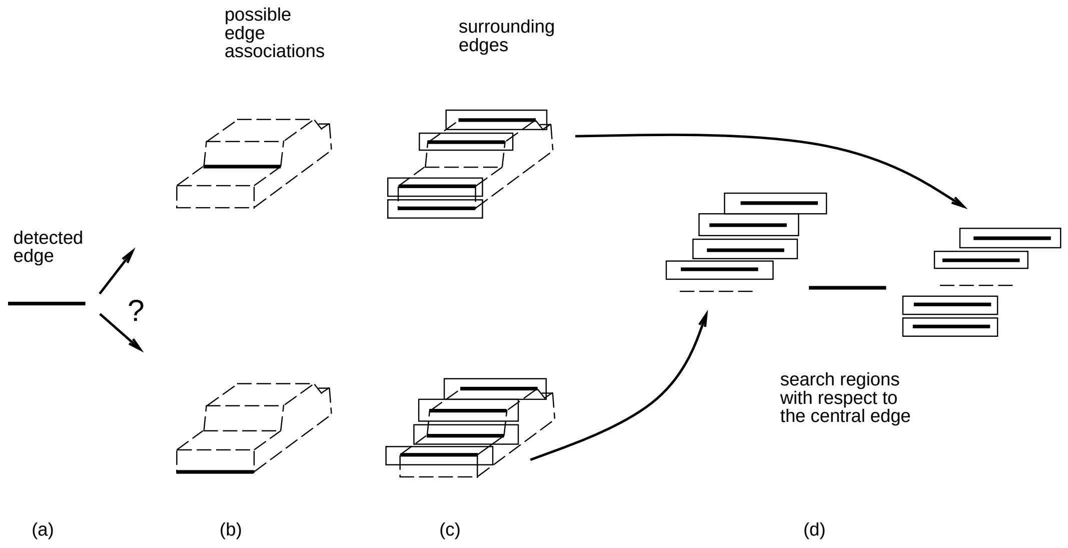

possible edge associations
surrounding edges
detected edge
?
search regions with respect to the central edge
(a)
(b)
(c)
(d)

Figure 4: (a) Once a detected edge is selected as the central edge the task is then to determine its identity (e.g. whether it is a bottom edge below the grill, the edge below the windshield, etc). (b) Each of the possible edge associations represents a hypothesis of a car model (here we presented only two such models). (c) The other edges of a car (surrounding edges), given the hypothesized identity of the central edge, are expected to be found within the specific local regions. (d) In order to support the hypothesis chosen in (b) the system looks only for specific edges (in terms of their sizes and orientations) within specific local regions.

We tested the performance of the system on several thousand video sequences. An example of a typical input image is shown in Figure 5. Once a system detects a still car it locks onto it (although it might fixate on different edges at different times) and the recognition is almost 100%. If the cars are moving and are separate from one another, the recognition accuracy is around 90%. However, when the cars become close to one another the recognition drops to about 70%, depending on how close the cars are and how much they are occluding each other. The system

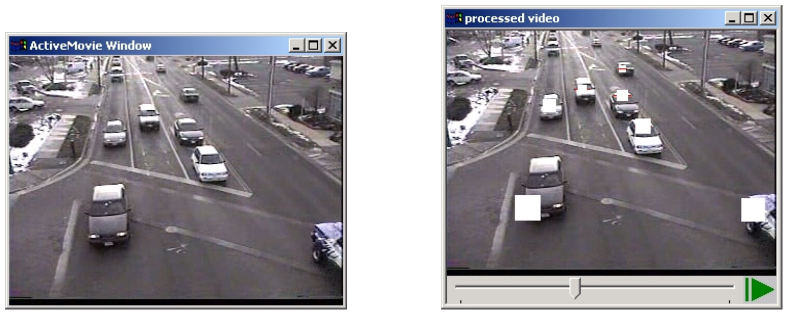


Figure 5: The left image illustrates the input to the recognition system while the right image is the result of the recognition process. Each white box denotes the region around the fixation point that contains an edge of high activation that was selected by the system. The location of the fixation point changes from frame to frame even for the same vehicle due to the noise, change in lighting conditions and occlusions.

mistakenly recognized a van as a car with about 30% confidence. It never substituted a pedestrian for a vehicle and locked onto side road clutter in less than 1% of the time. In addition to the car template we also implemented a van template. Although the vehicle template reduced the van/car confusion by about 40% it did not significantly increase the overall detection rate of the recognition system. The computational complexity associated with searching the space of horizontal and vertical edge activations is greatly reduced using selective attention thus allowing the system to process information in real time. However, introducing additional features such as edges of various directions proved to be computationally too expensive for the system to run in real time. Therefore, despite the fact that these edges can significantly improve the recognition rate it is unlikely that they can be employed without either dedicated hardware or specially designed fast extraction algorithms.

3 The Interactive Environment

In order to develop a system that can engage in bi-directional information exchange with the user, we constructed a specialized environment, a sort of virtual lab, that allows the user to interactively modify the parameters of the recognition system. Furthermore, the environment allows the user to even stop the video at any particular image and analyze it, edit parameters, collect statistics or label the objects and then either continue the video or advance it to the next frame. The environment consists of two modules: the player and the filter.

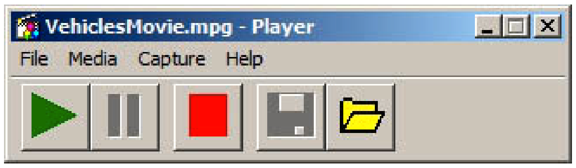
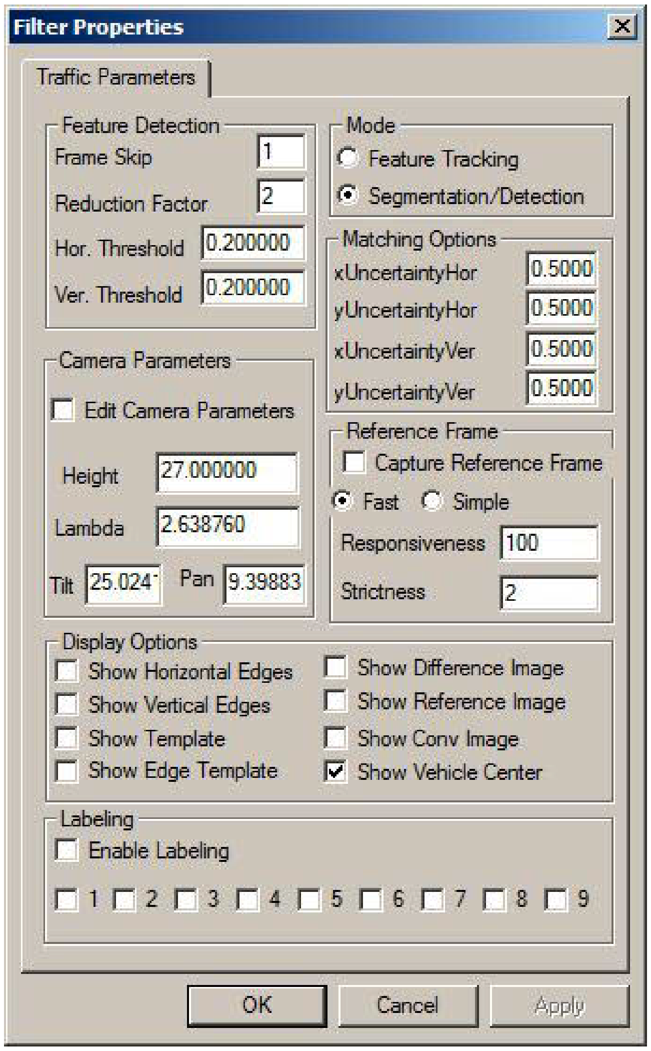


Figure 6: The player. Graphical representation of the program that allows the user to open prerecorded movie files or capture a real-time video using a simple web camera and then play, pause, stop, or seek to any point in the video.

3.1 The Player

The player is an independent program from the rest of the code base and it looks like a computer version of a VCR as illustrated in Figure 6. The user can open prerecorded movie files or capture a real-time video using a simple web camera and then play, pause, stop, or seek to any point in the video. The "single advance button", located to the right from the scrollbar, as shown at the bottom of the image in Figure 9, allows the user to advance video one frame at the time. In this way the user has time to analyze each image and this feature of the system is extremely useful during the labeling process. The player automatically determines the type of file (or real-time video stream), obtains the proper drivers and decoders in order to transform the source into a form that the vision filter understands. This ability allows it to run easily on almost any video format (e.g. MPG, AVI, ASF, etc) or using any video capturing hardware which is supported by MS DirectShow technology. The player also takes care of displaying the streams, with preview and processed display windows. The preview window displays the original (unprocessed) video while the processed window displays the result of one or more processing functions that can be selected in the properties panel (e.g. the locations and sizes of the extracted edges).
The player loads the filter and passes the video stream through it before being rendered to the screen. The filter is based on a MS DirectShow ActiveX transform technology and represents the interface between system and programmer. Once the video or real-time stream is playing, the user can modify the parameters of the vision filter by using a familiar windows properties dialog, Figure 7. This dialog is very special because it is actually part of the vision filter and not the player


Filter Properties

Traffic Parameters

Feature Detection
Frame Skip: 1
Reduction Factor: 2
Hor. Threshold: 0.200000
Ver. Threshold: 0.200000

Mode
Feature Tracking [✓ Segmentation/Detection]

Matching Options
xUncertaintyHor: 0.5000
yUncertaintyHor: 0.5000
xUncertaintyVer: 0.5000
yUncertaintyVer: 0.5000

Camera Parameters
Edit Camera Parameters
Height: 27.000000
Lambda: 2.638760
Tilt: 25.024
Pan: 9.39883

Reference Frame
Capture Reference Frame
[✓ Fast] Simple
Responsiveness: 100
Strictness: 2

Display Options
Show Horizontal Edges
Show Vertical Edges
Show Template
Show Edge Template
Show Difference Image
Show Reference Image
Show Conv Image
[✓ Show Vehicle Center]

Labeling
Enable Labeling
1 2 3 4 5 6 7 8 9

OK Cancel Apply

Figure 7: The properties window allows the user to interact with the recognition system and change various parameters while the system is running. The window is divided into several sections, each having different functionality. For example, the section ”Feature Detection” allows the user to set the threshold value that is used by the preprocessor during the edge extraction process while the section ”Mode” allows the user to choose whether the system should track the features or detect objects. One of the most useful features of the properties dialog window is that it can be used to collect various statistics such as the mean size and variance of vehicle edges vs. the non-vehicle edges. Similarly, the system can label objects that it tracks (or detects) and the user can then respond by telling the system whether the labeling is correct or not.

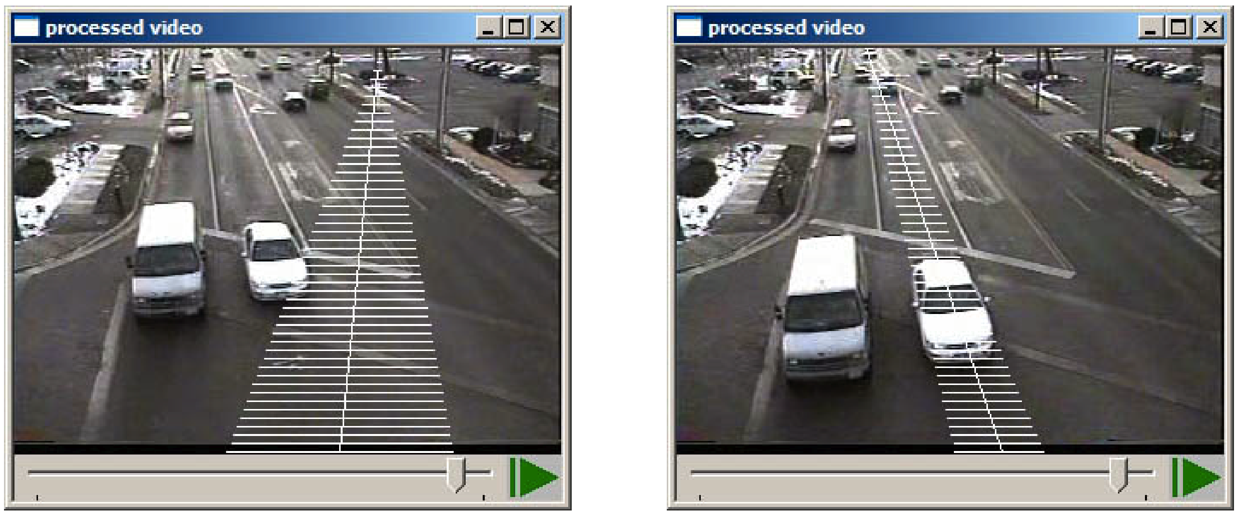


Figure 8: Projecting the direction of motion (the long straight line) and the expected car widths (a collection of small parallel lines) onto the processed image. The camera parameters can be adjusted until the direction of motion line is along the road and the lengths of small lines become good approximations of the car widths. The left image illustrates the wrong set of camera parameters while the right image shows a set of camera parameters that provides a good match between the expected direction and expected car widths.

application. From the programmer's prospective this means that he or she can modify the filter completely, even the interface, separately from the player. This saves time and allows the filter to be modified without having to recompile the player application.

3.2 The Properties Window

The properties window is divided into several sections, each having different functionality. In the following, we will briefly describe some of those functions. The section titled "Feature Detection" allows the user to set the following parameters: a) The number of frames that can be skipped. Although many video formats run at 30 frames per second, from our experiments processing every other frame does not reduce recognition rates. b) Size of the image that is being processed. The reduction factor of 2 means that the system is processing every other pixel. If the original image size is 340x240 then the reduction of 2 means that the system effectively processes the image size 170x120. c) The threshold values for the extraction of horizontal and vertical edges. Every edge with activation above the threshold value is selected as the potential edge whose membership has to be determined.
The section "Mode" allows the user to choose between the tracking and detection mode. In the "Feature Tracking" mode the system tracks selected features using contextual information coming from surrounding features. In the "Segmentation/Detection" mode the system performs image segmentation by detecting the vehicles and classifying them into one of the predetermined categories,

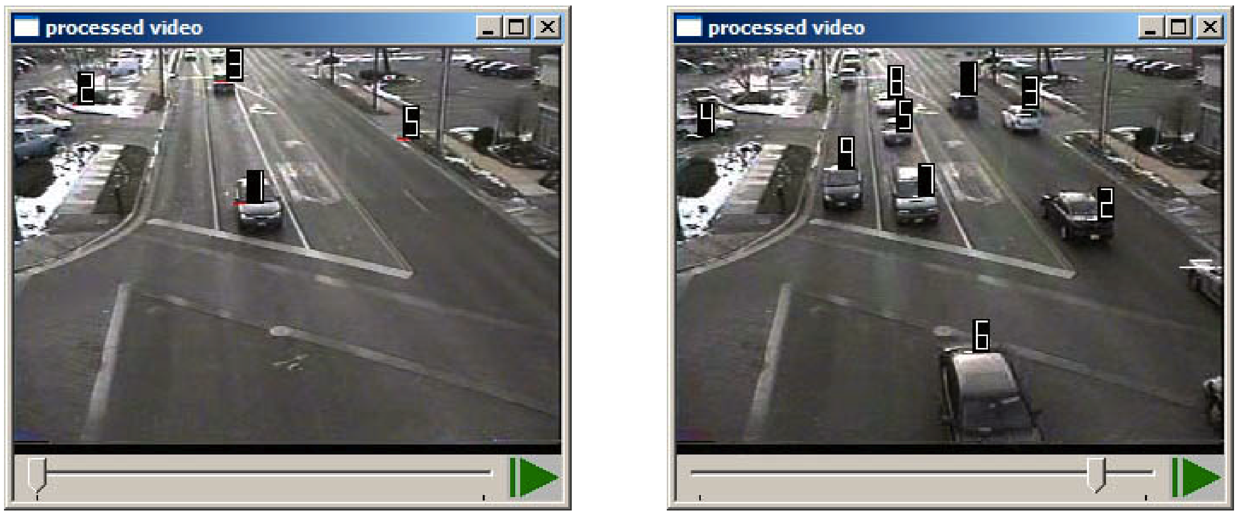


Figure 9: Left image: Some of the most prominent edges that were extracted by the system and labeled with numbers where the number 1 corresponds to the most prominent edge and number 4 to the least prominent edge. Right image: The system assigns a number to each edge that is classified as a "vehicle edge" which means that a number corresponds to a vehicle being detected. The number 1 corresponds to the edge that is associated with one of the vehicle edges with highest confidence whereas the highest number corresponds to the edge/vehicle detected with lowest confidence. Notice that the system can detect the vehicles regardless of their direction of motion.

e.g. car, vans, trucks, etc.
The size of the region within which the center of the specific edge is expected to be found can be specified in the section "Matching Options". This size is calculated as a percentage of the size of the edge itself, which means that the location of the center of a longer edge would be associated with larger uncertainty. The section "Reference Frame" allows the user to choose the method for calculating the difference image. By checking the button "Fast" the system calculates the difference frame by subtracting the current frame from the previous frame and by checking the button "Simple" the system calculates the difference image by subtracting the current image from the background image that should contain no vehicles.
Camera parameters, such as zoom, tilt, pan and camera height are controlled within the "Camera Parameters" section. This gives a great flexibility to the system and it means that any camera, regardless of its location with respect to the road, can be easily calibrated. In order to verify how good are the chosen estimates of the camera parameters we designed two ways of visual verification. First, the user can choose to project on the image the expected direction of the road, which is in Figure 8 drawn as a long line going from bottom to top of the image. Second, the user can project on the image, at any location, the expected widths of the cars and then visually compare the widths of real cars to the drawn lines representing expected widths. The image on the left in Figure 8

shows an arbitrary set of camera parameters while the parameters used for the image on the right show good match between expected direction of the road and car widths.

3.3 Labeling

One of the most useful features of the properties dialog window is contained in the ”Labeling” section. If this feature is enabled, the system displays on the processed window the numbers with which it labels the object that are analyzed and thus allows the user to determine how many of the labeled objects are correctly labeled. For instance, if the goal is to collect the statistics of the correctly tracked edges then the objects are edges and in boxes 1 to 9 the user puts a check mark every time a specific edge (with a specific number) is the correct edge. One example is presented in Figure 9 where only the edges labeled 1 and 3 belong to vehicles are therefore only the boxes and 1 and 3 should be checked out. The user can then advance the video for one frame and repeat the labeling procedure or choose completely different position within the video to continue with the labeling process. In addition to gathering statistics about the number of correctly and incorrectly labeled edges, the user can also collect more detailed information such as the average size and intensity of correct/incorrect edges and their variances. We used this technique in order to determine which features are the most appropriate for vehicle description.

4 Tracking Object Features Using Dynamically Defined Spatial Context

It is well known that context plays an important role in processing visual information during various recognition tasks. For example, human perception of an edge is so influenced by context that we sometimes "see" edges where there are none. This is due to the fact that information contained in the local region is often either ambiguous or insufficient for reliable interpretation of that region. On a cellular level, it has been shown that neuronal responses are modulated by the context within which a stimulus is presented and that a single neuron is capable of integrating information over large areas of visual space (Gilbert et al., 2000).

Compared to recognition tasks, the investigation of the role of the spatial context in tracking tasks received much less attention. Instead, much more importance has been placed on the role of temporal context that has usually been modeled using the Kalman filter method (Kalman, 1960). Another use of contextual information has been suggested by Intille et al (Intille et al., 1997) where context has been defined as knowledge about the objects that are being tracked and their mutual relationships.

Research related to tracking problems has been extensively studied within the field of computer vision (Faugeras, 1993). Tracking algorithms found applications in various domains such as: traffic monitoring (Coifman et al., 1998), video surveillance and monitoring (Lipton et al., 1998b), and analysis of video streams obtained from airborne observers (Cohen and Medioni, 1998). Commonly used approaches in tracking applications are the 3D and 2D model-based approaches (Lowe, 1992). Despite the fact that the 3D motion recovery from 2D images is often an ill-posed problem, the 3D model-based approaches are very successful in many applications, such as human (Gavrila and Davis, 1996) and vehicle (Koller et al., 1993a) tracking. One of the strengths of 3D models is that they can predict self-occlusions and self-collisions. The serious weakness, on the other hand, is their reliance on detailed geometric models, which are unavailable for a large number of objects. Furthermore, detailed geometric description is often unnecessary since once the object is detected, tracking can be achieved by tracking just a portion of the object.

Feature-based approaches (Bretzner and Lindeberg, 1998), can easily deal not only with self-occlusions but also with partial occlusions coming from any other object as long as some of the features remain visible. If the final goal is to track an object, then the problem is how to group distinct features such that they represent the same object (Coifman et al., 1998). Another problem is related to feature selection. The simpler the feature is, the larger the class of objects that contain the feature, and detection of such a feature is computationally less involved. However, the very fact that the structure of the feature is simple means that there can be multiple matches. This is commonly known as the correspondence problem. One way of reducing this ambiguity associated with feature matching is to increase the complexity of the feature, and in the limit a whole

object can be considered as a complex feature (von der Malsburg, 1999). Another possibility is to introduce the context within which a feature is detected without increasing the complexity of the feature; that is the approach we take in this paper.
In (Wang et al., 2004), we proposed a model for tracking features using a dynamically defined spatial context. More specifically, the spatial context consists of a collection of features within the local neighborhood around the feature that is being tracked. The model is inspired by the role contextual information plays during perception and processing of information on the neuronal level (Gilbert et al., 2000). We applied the model to tracking horizontal and vertical edges extracted from real-time video streams that contain moving vehicles. Since horizontal and vertical edges are very prominent features in vehicles and vehicles are rigid objects, feature tracking, in this application, is equivalent to object tracking.

4.1 Implementation

The tracking procedure starts by selecting the most prominent edges from the image. Each of these edges becomes the central edge that is to be tracked. Given the location of the central edge, the system then selects a group of surrounding edges to provide a contextual support to the central edge. In the current implementation, this group consists of four surrounding edges. Since the boundaries of the object (whose local region the system is tracking) are not known, the supporting features might not belong to the object. For example, one or more features could belong to another object or to the background. This means that without knowing where the object is within a scene, the context has to be defined dynamically. Initially, the supporting features are just hypothesized and then if they continue to appear in successive frames the hypothesis that they belong to the same object is confirmed. Otherwise, those features are discarded and in their place another group of features is selected.
If the central edge has just been acquired, the system explores the local neighborhood around the previous location of the central edge for possible matches. Otherwise, if the central edge has already been tracked, the system calculates the probable location of the central edge and examines the local neighborhood around this location for possible matches. In order to predict the location of the central edge, we implemented the Kalman filter using a constant velocity model. However, it should be noted that the Kalman filter did not significantly improve the results, but mostly served the purpose of reducing the search time by constraining the area the system explored in search of the location of the central edge.
In general, the system finds several possible matches for each central edge. The next step is then to use contextual edges to select the best match. For each tentative location of the central edge, the system searches for supporting edges by investigating the local neighborhoods around the expected locations of the supporting edges. The similarity between the selected supporting edge and the previous supporting edge is calculated as the absolute difference of their activation values. The

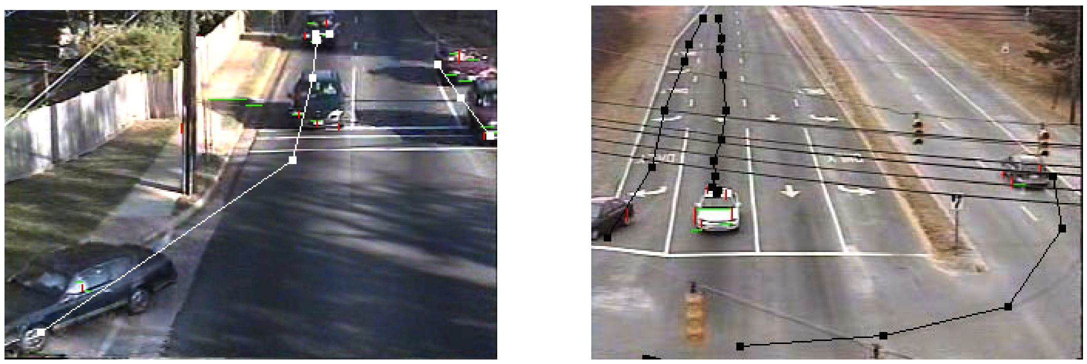


Figure 10: The figure on the left illustrates the robustness of the tracking system to the changing lighting conditions and the changing orientation of the object whose feature the system is tracking. Although the system tracked the same object, it didn't track the same feature since the feature that it initially locked onto disappeared in subsequent frames. The figure on the right shows that the system can track not only moving features but also the ones that come to a stop since matching is based on spatial context.

similarity between the tentative central edge and the previous central edge is calculated in the same way. The contribution from the supporting edges, the support function, is calculated as the sum of their individual contributions.
To prevent the system from tracking the same group of edges, once an edge is being tracked the supporting edges only serve as its context and are not tracked independently. In this way, the system can assign its computational resources to other areas within the scene.

4.2 Results

The system is based on the MS DirectShow ActiveX transform technology and is implemented on a standard PC with Intel Pentium 700MHz chip under Windows 2000 operating system. We have tested the performance on video streams captured by a standard CCD camera and a Web Cam under various lighting and traffic conditions. Our tests show that the system is robust to luminance changes and can easily handle congested traffic scenes.
It is very difficult to quantify the number of correctly tracked edges since it is not always possible to visually detect slight changes in the edge location within a vehicle and thus the edge identity. Furthermore, it often happens that the feature that the system is tracking disappears from the scene because the object changes its orientation with respect to the camera, as shown in Figure 2. Therefore, instead of edges, we choose to count the number of vehicles that are correctly tracked. We classify a vehicle as being correctly tracked if it is continuously tracked from the time it is first acquired by the system until it disappears from the scene. The performance rate on the tested 172 vehicles is 90.1%. It should be emphasized that our tests mostly involve traffic scenes taken at intersections where vehicles make a turn or come to a stop and start again.

Table 1: Percentage of correctly tracked vehicles for different numbers of supporting edges.

performance rate	number of supporting edges
86.6 %	0
87.2 %	2
90.1 %	4

The performance of the system is not significantly reduced if the Kalman filter is not used to estimate the locations of the central edges. The performance rate obtained on the same 172 vehicles without using the Kalman filter is 84.9%. The reason is that introduction of the contextual support for each central edge effectively increases the discrimination power of the central edge among tentative central edges. Thus, it makes it possible for the system to reliably locate the best match for the central edge within a larger neighborhood around its location in the previous time frame. Therefore the model that we propose provides a complimentary and alternative method for tracking. It can be combined with the Kalman filter-based method, as we did, to provide better results of tracking.

In order to investigate how the contextual information influences the performance of the system, we tested the tracker using different number of supporting features ranging from 0 to 4. Table 1 gives the performance rates of our tracker for different numbers of supporting features. The results were obtained on the same video sequence as before and the Kalman filter was used in all the three cases. Although the performance rate of the tracker has a high variance, it is clear that by increasing the number of supporting features, the system becomes more robust and makes fewer mistakes.

5 Partitioning a Feature Space Into Locally Confident Regions

5.1 Introduction

When a human teaches a student to recognize objects, and the student is not able to recognize an example of a given class, he or she can communicate that to the teacher during the learning process. Although most supervised learning algorithms are inspired by human teaching processes, it is difficult, if not impossible, to find algorithms that can engage in such interactive learning. Recognition systems based on some of the most prominent learning algorithms, such as back-propagation, radial basis function (RBF), support vector machine (SVM) and hidden Markov model (HMM) always "learn" during the training phase by updating weights or other parameters of the system. But isn't it better for a recognition system to say that it can not learn an object category if the features that the teacher choose to represent objects are not sufficient for class separation? Because in this case there is no point in learning but instead the teacher should provide a better representation of the object.
In order to achieve this goal, to build a system that can interactively engage in the exchange of information during the learning process, the system has to know its limitations and when to ask a question. In this work we present a first step toward constructing such a system. Given a confidence level, that can be set by a user/teacher, we constructed a system that learns object categories by partitioning the feature space into regions within which it has above threshold confidence of correctly classifying a pattern and into regions with confidence below a pre-set threshold value. In this way, the user/teacher can help the system by providing additional information (e.g. by introducing new features) only for selected low-confidence regions. But perhaps more importantly, the user knows the limitations and strengths of the system - how well the system has learned the task before it is used on unseen patterns.

5.2 Network architecture and implementation

In this section we present the architecture of the network that partitions the feature space into locally confident prototype regions.
The network consists of three layers: the input layer, the hidden layer and the output layer. Each input layer unit represents an attribute of the incoming feature vector. All units of the input layer are connected to all the units of the hidden layer. Each unit of the hidden layer, called a prototype unit, is class specific and therefore connected to only one unit of the output layer. A prototype unit represents a local region characterized by its weight vector (the center of the region) and its size (the radius of the hyper sphere). A prototype unit outputs the confidence level over the region it represents.
The learning algorithm of the network consists of two phases. The task of the first phase is to associate with each sample point a hyper sphere with the radius equal to the distance to the closest

point of the opposite class. For each such hyper sphere, we calculate its confidence measure as described in (Wang et al., 2003). As a result, there are as many regions as there are sample points. The goal of the second phase is to reduce the number of regions. If the point is covered by more than one region, then it is assigned to the region with the highest confidence. Regions constructed in this way we call prototypes.
The feature space is thus divided into three types of regions. Those that represent only one class, the regions that represent more than one class and the regions that do not belong to any class - the holes in the feature space. If the point falls in the region of the first type, then the assignment is unambiguous, provided that the confidence level of the region is satisfactory. The regions of the second and third type represent the areas for which the network does not have sufficient knowledge for reliable classification. In order to compare our results with other methods, we choose to assign samples from those regions to the prototype with the smallest distance-to-size ratio, where the distance refers to the distance from the sample point to the center of the prototype.

Table 2: Results on the Breast Cancer Dataset

Method	Accuracy%
FSM	98.3
LCN	98.0
2NN	97.1
21NN	96.9
CART	96.0

Table 3: Results on the SatImage Dataset

Method	Accuracy%
LCN	90.6
k-NN	90.6
LVQ	89.5
DIPOL92	88.9
RBF	87.9

Table 4: Results on the Letter dataset

Method	Test Error%
Alloc80	6.4
k-NN	6.8
LCN	7.4
LVQ	7.9
Quadisc	11.3

5.3 Properties of the system and results

An important property of the system is that it knows its limitations since its knowledge is represented locally. During the training phase, this property allows the system to interact with a teacher that can provide more samples for the critical region of the feature space or introduce new features for object description. Again, this can be done only on local areas of the feature space and would not adversely affect the regions over which the system has already acquired knowledge. Similarly, during the testing/recognition phase, this property of local representation of knowledge, allows the user to provide assistance to the system for the examples that happen to fall at the intersection of regions of different classes or in the regions not covered by prototypes. In this way the network presents the first step in constructing a more interactive recognition system.
In comparison to Nearest neighbor rule (Cover and Hart, 1967) and RCE network (Reilly et al., 1982; Scofield et al., 1987), our approach offers several advantages: it is computationally efficient, it is optimal in the Bayesian sense and there are no external parameters associated with proto-

type regions. In addition, the algorithm for constructing prototype regions is extremely easy to implement and offers a unique partitioning of the feature space regardless of the order in which the training samples are presented to the system.
We tested our algorithm on three Statlog datasets from the StatLog repository(Michie et al., 1994): the Breast Cancer, SatImage and Letter datasets. For each of the three datasets, our results are compared with those of the four best algorithms reported and summarized in Tables 1 - 3.
The results have shown that our method performs consistently well on real world datasets. It should also be emphasized that in addition to the test error we can obtain much more detailed information about the performance of the network. Take the SatImage dataset as an example. Among the 2000 test samples, only 93 (4.65%) are covered by wrong prototypes, 204 (10.2%) are covered by both right and wrong prototypes and 89 (4.45%) are not covered by any prototype at all. In many situations, like the breast cancer diagnosis, it is extremely important not to miss any case of cancer. Therefore, an important property of the system is to know its limitations and not to be used for the regions where its knowledge is insufficient for reliable classification but instead consult a doctor or other expert system.

6 Neighborhood Size Selection in the k-Nearest Neighbor Rule

6.1 Introduction

One of the most attractive classification algorithms, first proposed by Fix and Hodges in 1951, is the nearest neighbor rule (Fix and Hodges, 1951). It classifies an unknown pattern $\vec{X}$ into the class of its nearest neighbor in the training data. Geometrically, each labeled observation in the training dataset serves as a prototype to represent all the points in its Voronoi cell.

It can be shown that at any given point $\vec{X}$ the probability that its nearest neighbor $\vec{X}'$ belongs to class ω_i converges to the corresponding a posteriori probability $P(\omega_i|\vec{X})$ as the number of reference observations goes to infinity, i.e., $P(\omega_i|\vec{X}) = \lim_{n\to\infty} P(\omega_i|\vec{X}')$. Furthermore, the asymptotic probability of error L_{NN} of the nearest neighbor rule is bounded by

$$L^* \leq L_{NN} \leq L^*(2 - \frac{M}{M-1}L^*), \tag{9}$$

where L^* is the optimal Bayes probability of error. Therefore, the nearest neighbor rule, despite its extreme simplicity, is asymptotically optimal when the classes are non-overlapping. However, when the classes are overlapping, the nearest neighbor rule is suboptimal. The problem occurs at overlapping regions where $P(\omega_i|\vec{X}) > 0$ for more than one class ω_i. In those regions the nearest neighbor rule deviates from the Bayes decision rule by classifying $\vec{X}$ into class ω_i with probability $P(\omega_i|\vec{X})$ instead of assigning $\vec{X}$ to the majority class with probability one.

In principle, this shortcoming can be overcome by an obvious extension, the k-nearest neighbor rule. As the name suggests, this rule classifies $\vec{X}$ by assigning it to the class that appears most frequently among the k nearest neighbors. Indeed, as shown in (Stone, 1977; L. Devroye and Lugosi, 1994), the k-nearest neighbor rule is universally consistent provided that the speed of k approaching n is properly controlled, i.e., $k \to \infty$ and $k/n \to 0$ as $n \to \infty$. However, choosing the optimal value k in a practical application is always a problem, due to the fact that only a finite amount of training data is available. This problem is well known as the bias/variance dilemma in the statistical learning community (Geman et al., 1992). In practice, one usually uses methods such as cross-validation to pick the best value for k.

In our recent work (Wang et al.,) we addressed the problem of neighborhood size selection. In the k-nearest neighbor rule the value of k, once determined by minimizing the overall probability of error, is the same for all points in the feature space. However, the optimal value of k at one point does not have to be, and in general is not, the same as the optimal value of k at some other point. The value of k should therefore be determined locally. The question is: what criterion should one choose to determine the optimal value of k?

Our approach to neighborhood size selection is based on the following observation: There is always a certain non-zero probability that a decision is wrong when it is made from a finite number of observations. Therefore, it is advantageous to know what the probability of error is when making a decision and to keep this error under control. For example, in many applications, such as in medical diagnosis, the confidence with which a system makes a decision is of crucial importance. Similarly, in situations where there are several classes, not every class has to be of the same importance and each class might require a different level of decision confidence. Instead of requiring a fixed value of k throughout the feature space, in such applications it is more important to fix the confidence level. Based on this observation, we proposed a method that dynamically adjusts the number of nearest neighbors until a satisfactory level of confidence is reached.

6.2 Results on Real-World Datasets

In this section we present the results of our algorithm that has been tested on several real-world datasets from the UCI Machine Learning Repository (Blake and Merz, 1998). We used the leave-one-out method to estimate the classification error of the confident-nearest neighbor (Confident-NN) rule and the k-nearest neighbor (k-NN) rule. For each dataset, the lowest classification error achieved by the k-NN rule, together with the corresponding k value (shown in parenthesis), is reported in Table 5. Similarly, we report the lowest error rates obtained by the confident-nearest neighbor rule. The corresponding confidence level and the average number $\bar{k}$ of the nearest neighbors used are given in parenthesis.

Table 5: Comparison of results for the k-NN and Confident-NN rules

Dataset	k-NN (k)	Confident-NN (CFD;$\bar{k}$)
BreastCancer	0.0249 (5)	0.0278 (75%;2.1)
Ionosphere	0.1339 (1)	0.1339 (70%;1.0)
Liver	0.3043 (9)	0.3130 (90%;19.9)
Pima	0.2396 (19)	0.2461 (90%;20.0)
Sonar	0.1731 (1)	0.1683 (75%;2.5)

Although the error rates of the k-nearest neighbor rule are slightly better compared to the error rates of the confident-nearest neighbor rule, there are several advantages of the confident-nearest neighbor rule which we would like to emphasize. First, in many applications, such as medical diagnosis, the overall error is not the only or the most important goal. Instead, the confidence with which a decision is made is of crucial importance. Second, the decision confidence is not always the same for all the regions of the feature space. For example, when diagnosing a specific disease, a subset of some symptoms and their values might be more ambiguous than other symptoms, and that should be reflected in the decision-making process. Finally, not all the classes require the same confidence level. For example, the decision whether a given set of symptoms indicates that a patient has the flu or lung cancer is of different importance, and the confidence in diagnosing lung cancer should be much higher when compared with the flu. The original k-nearest neighbor

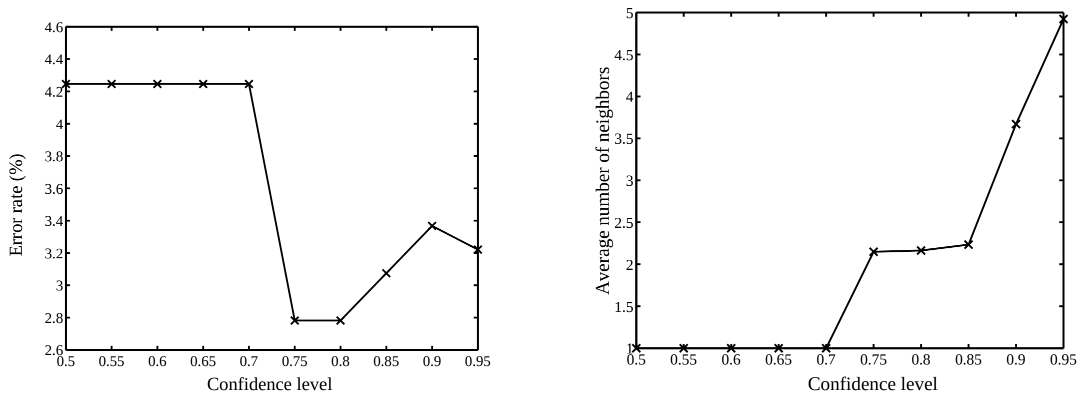


Figure 11: Left: Leave-one-out estimate of the classification error at different confidence levels. Right: Average number of neighbors used at different confidence levels.

rule, since it is based on minimizing the overall error rate, does not address these issues. The confident-nearest neighbor rule, on the other hand, provides a locally defined statistical measure of the confidence level and therefore can dynamically adjust the number of nearest neighbors until all these requirements are satisfied.

To show how the error rates of the confident-nearest neighbor rule change as a function of the confidence level, we use the data from the Wisconsin Breast Cancer dataset, as illustrated in Fig. 11 (left). It is important to notice that the error rate does not necessarily decrease as the confidence level increases. Using the confidence criterion, the lowest error rate of 0.0278 is achieved at confidence level $75\% - 80\%$. It is slightly higher than the lowest error rate of 0.0249, which is obtained by the k-nearest neighbor rule. However, the lowest error rate of the k-nearest neighbor rule, although slightly better than the one given by the confident-nearest neighbor rule, does not contain any information regarding the confidence with which the decision is made.

Table 6: Mean confidence levels for the misclassified and correctly classified data points

Dataset	Misclassified (S. D.)	Correctly Classified (S. D.)
BreastCancer	0.8581 (0.0068)	0.8723 (0.0005)
Ionosphere	0.7500 (0.0000)	0.7500 (0.0000)
Liver	0.9231 (0.0013)	0.9269 (0.0009)
Pima	0.9244 (0.0010)	0.9291 (0.0005)
Sonar	0.8518 (0.0057)	0.8652 (0.0020)

Although the number of nearest neighbors used in the confident-nearest neighbor rule varies from point to point, in Fig. 11 we plot the average number $\bar{k}$ of nearest neighbors at different confidence levels. As one can see, more neighbors are used as one increases the confidence requirement. However, combining both graphs in Fig. 11, it follows that more neighbors do not necessarily result in lower error rates. This is because as we increase the confidence level requirement, more and more neighbors are needed to satisfy the higher confidence level requirement. However, we are only given a limited number of training samples and are not allowed to draw more samples from the close neighborhood of the query points, therefore training samples that are further away from the query points are used as neighbors and they may have different a posteriori probability distributions.

In the most general pattern classification scenario, classes may overlap in some regions of the feature space. Without knowledge of the underlying distribution, it is hard to tell whether a training point is actually misclassified by the classifier, or whether the data point itself is not labeled as the true majority class. We compute the mean confidence level and the mean neighborhood size of the misclassified data and the correctly classified data separately, to see whether they have different characteristics. The results are reported in Tables 6 and 7 respectively. The numbers in the parenthesis (S. D.) are the corresponding standard deviations. As one can see, on average, the

Table 7: Mean neighborhood size for the misclassified and correctly classified data points

Dataset	Misclassified (S. D.)	Correctly Classified (S. D.)
BreastCancer	3.26 (0.67)	2.12 (0.03)
Ionosphere	1.00 (0.00)	1.00 (0.00)
Liver	25.48 (4.73)	17.34 (2.55)
Pima	33.51 (4.57)	15.55 (1.54)
Sonar	2.94 (0.27)	2.38 (0.08)

misclassified data tend to have lower confidence levels and use more neighbors for training, which means that even with more points to learn from, poorer confidence levels are nonetheless achieved on the misclassified data. This is a clear indication that the misclassified data is really lying in the regions where two different classes overlap, therefore attempts to further reduce the classification error may run into the risk of over fitting the training data.

Table 8: Trade-off between the rejection rate and error rate

Confidence Level	Reject Rate	Classification Error Rate
50%	0	0.0249
60%	0	0.0249
70%	0.0337	0.0167
80%	0.0337	0.0167
90%	0.0981	0.0081
95%	0.0981	0.0081

In many applications, wrong classifications are costly. Therefore, instead of trying to make a decision when the confidence is low, a better alternative would be to reject those patterns with low confidence levels and make a classification only when the confidence level is high. Since the misclassified data tend to have lower confidence levels compared to the correctly classified data, rejecting patterns with low confidence levels will lead to a reduction in the error rate on the remaining data. The rejected patterns should be set aside until further information allows a satisfactorily confident decision to be made. For the Breast Cancer dataset, the lowest error rate is achieved by the k-nearest neighbor rule with k set to 5. In the k-nearest neighbor rule, we reject the patterns whose confidence levels from their 5 nearest neighbors are below a specific confidence level. The rejection rate and the error rate on the remaining data for a range of different confidence levels are reported in Table 8. As easily seen, the reject rate increases monotonically with the confidence level and in the meantime the error rate on the remaining data is decreasing. At the 95% confidence level, a recognition rate of 99% is achieved with a reject rate less than 10 percent.

7 Summary of the Most Important Results

In summary, within the period of September 1st 2001 and September 1st 2004, we succesfully completed the following projects:

• Constructed a working system for detection and tracking of moving or still vehicles from real-time video streams. In constructing an artificial recognition system for real-time processing of video streams we draw inspiration from the way the human visual system analyzes complex scenes. The properties of the human visual system that we utilize in our system are: representation of objects in terms of parts, selective attention, saccadic eye movements and hierarchical processing of visual information. During the recognition process, the system efficiently searches the space of possible segmentations by investigating the local regions of the image in a way similar to human eye movements, probing and analyzing different locations at different times. The computational complexity associated with searching the space of edge activations is greatly reduced using selective attention, thus allowing the system to process information in real time using a regular Pentium III, 700MHz processor. The system was tested on real-world data sets and exhibits very high recognition rates. The fact that we use feature-based and feature-centered object representation allows translation invariant recognition and makes the system very robust to occlusions. Similarly, the system can easily deal with variable lighting conditions since the features are edges, and their extraction is not affected with overall change in illumination.

• Developed an algorithm for adaptive sensory processing. The algorithm provides a solution to two of the most significant problems related to edge extraction: 1) selection of the scale and threshold value, and 2) efficient use of computational resources. Although the optimal threshold and scale values in general depend on local information, in most applications these values are set a-priori or globally. Our approach, on the other hand, uses contextual information to dynamically, and thus locally, determine these two parameters. In addition, the algorithm efficiently uses computational resources by extracting edges only from the local regions as opposed to processing the whole image.

• Built an environment for image acquisition and image processing that allows users to interactively modify the parameters of the recognition system while the system is running. The environment allows the user to even stop the video at any particular frame and analyze it and then either continue the video or advance it to the next frame. The bi-directional exchange of information between the system and the user is facilitated through a specially constructed dialog window. This window allows the user, among other things, to: a) edit and adjust the camera parameters, b) select the mode for system operation by choosing between a tracker or a detection mode, c) select the method for calculating the background image, and d) label the objects/features, or verify whether the system correctly labeled them, and collect various statistics.

• Developed a model for tracking object features using a dynamically defined spatial context. More specifically, the context is defined as a collection of features within the local region, surrounding the feature that is being tracked. The model is inspired by the role contextual information plays during perception and processing of information on the neuronal level. The model does not rely on any knowledge about the object, and therefore the collection of contextual features in one frame is just a hypothesis that is then reconfirmed or rejected in subsequent frames. We applied the model to tracking horizontal and vertical edges extracted from real-time video streams that contain moving vehicles. Our results show that the system can successfully utilize contextual information since the performance improves as the number of contextual features increases.

• Developed a new learning algorithm that partitions the feature space into local regions, and is capable of communicating to the user/teacher the confidence with which it recognizes an object. We also constructed a network that implements this algorithm and tested the network on real-world data. The results demonstrate that the network performs consistently well and favorably compares to some of the best learning algorithms. In addition, our algorithm offers significant advantages over the Reduced Coulomb Energy (RCE) rule in that there are no external parameters associated with prototype regions when compared to RCE rule. The algorithm for constructing prototype regions is extremely easy to implement and offers a unique partitioning of the feature space regardless of the order in which the training samples are presented to the system.

• Developed a new method for neighborhood size selection in the k-nearest neighbor (k-NN) rule that is based on the concept of statistical confidence. The new algorithm is tested on several real-world datasets and yields results comparable to the k-nearest neighbor rule. However, in contrast to the k-nearest neighbor rule, that uses a fixed number of nearest neighbors throughout the feature space, our method locally adjusts the number of nearest neighbors used until a satisfactory level of confidence is reached. This distinct property of our method can be of crucial importance in some applications, such as in medical diagnosis, where the confidence with which a decision is made is equally or more important than the overall error rate. One of the very important results of this work is the realization that for a given training dataset, and a given level of confidence, there is always a limit in reducing the error rate. However, if one wants to further reduce the overall error rate and keep the same confidence level in making decisions, one can still do it, but at the expense of reducing the size of the feature space over which decisions can be made.

8 List of publications

Papers published in peer-reviewed journals, books or conference proceedings:

J. Wang and P. Neskovic and L. N. Cooper, ”Context-based Tracking of Object Features”, IJCNN, 2004.

J. Wang and P. Neskovic and L. N. Cooper, ”Tracking Object Features Using Dynamically Defined Spatial Context”, ECOVISION, 2004.

P. Neskovic, D. Schuster and L. N Cooper, ”Biologically inspired recognition system for car detection from real-time video streams”, Neural Information Processing: Research and Development, J. C. Rajapakse and L. Wang (eds.), Springer - Verlag, pp. 320-334, 2003.

P. Neskovic and L. N. Cooper, ”Providing context for edge extraction in application to detection of cars from video streams”, International Conference on Engineering Applications of Neural Networks, Malaga, 2003.

J. Wang and P. Neskovic and L. N. Cooper, ”Partitioning a feature space using a locally defined confidence measure”, Joint 13th International Conference on Artificial Neural Networks and 10th International Conference on Neural Information Processing, Istanbul, 2003.

P. Neskovic and L. N. Cooper, ”Combining ”what” and ”where” information within a fixation and between fixations”, ICCNS, 2003.

P. Neskovic, D. Schuster and L. N. Cooper, ”A recognition system that uses saccades to detect cars from real-time video streams”, ICONIP, 2002.

Manuscripts submitted but not published:

J. Wang and P. Neskovic and L. N. Cooper, ”Neighborhood Size Selection in the k-Nearest Neighbor Rule Using Statistical Confidence”, *submitted to journal Pattern Recognition.*

Submitted inventions:
P. Neskovic and L. N. Cooper, ”A model for identification of objects in visual scenes that integrates available information using Bayesian inference”, Provisional Patent Application, March 11, 2004, Application Number 60/552,330.

References

Agarwal, S. and Roth, D. (2002). Learning a sparse representation for object detection. In *7th European Conference on Computer Vision*, pages 113–130.

Bell, A. J. and Sejnowski, T. J. (1997). The independent components of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338.

Biederman, I. (1987). Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94:115–147.

Blake, C. L. and Merz, C. J. (1998). Uci repository of machine learning databases. Technical report, Dept. of Information and Computer Sciences,University of California, Irvine. URL `http:://www.ics.uci.edu/ mlearn/MLRepository.html`.

Bretzner, L. and Lindeberg, T. (1998). Feature tracking with automatic selection of spatial scales. *Computer Vision and Image Understanding*, 71:385–392.

C. Goerick, N. D. and Werner, M. (1996). Artificial neural networks in real-time car detection and tracking applications. *Pattern Recognition Letters*, 17:335–343.

Cohen, I. and Medioni, G. (1998). Detection and tracking of objects in airborne video imagery. In *Interpretation of Visual Motion Workshop, CVPR.*

Coifman, B., Beymer, D., McLauchlan, P., and Malik, J. (1998). A real time computer vision system for vehicle tracking and traffic surveillance. *Transportation Research*, C 64(6):271–288.

Cover, T. M. and Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27.

E. Wachsmut, M. O. and Perrett, D. (1994). Recognition of objects and their component parts: responses of single units in the temporal cortex of the macaque. *Cerebral Cortex*, 4:509–522.

Faugeras, O. (1993). *Three Dimensional Computer Vision.* MIT Press.

Fix, E. and Hodges, J. (1951). Discriminatory analysis, nonparametric discrimination: consistency properties. Tech. report 4, USAF School of Aviation Medicine, Randolph Field, Texas.

Gavrila, D. M. and Davis, L. S. (1996). 3-d model-based tracking of humans in action: a multi-view approach. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 73–80.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58.

Gilbert, C., Ito, M., Kapadia, M., and Westheimer, G. (2000). Interactions between attention, context and learning in primary visual cortex. *Vision Research*, 40(10-12):1217–1226.

Gonzales, R. and Woods, R. (1993). *Digital Image Processing.* Addison-Wesley Publishing Company.

Hubel, D. H. and Wiesel, T. N. (1968). Receptive fields and functional architecture of monkey striate cortex. *J. Physiol*, 195:215–244.

Intille, S., Davis, J., and Bobick, A. (1997). Real-time closed-world tracking. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 697–703.

Kalman, R. F. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 1:35–45.

Keller, J., Rogers, S., Kabrisky, M., and Oxley, M. (1999). Object recognition based on human saccadic behaviour. *Pattern Analysis and Applications*, 2:251–263.

Koller, D., Daniilidis, K., and Nagel, H.-H. (1993a). Model-based object tracking in monocular image sequences of road traffic scenes. *International Journal of Computer Vision*, 10(3):257 281.

Koller, D., Danilidis, K., and Nagel, H. (1993b). Model-based object tracking in monocular image sequences of road traffic scenes. *International Jouran of Computer Vision*, 10(3):257–281.

Koller, D., Weber, J., and Malik, J. (1994). Robust multiple car tracking with occlusion reasoning. In *Proceedings 5th European Conference on Computer Vision*, pages 189–196.

L. Devroye, L. Gyorfi, A. K. and Lugosi, G. (1994). On the strong universal consistency of nearest neighbor regression function estimates. *Annals of Statistics*, 22:1371–1385.

Lipton, A., Fujiyoshi, H., and Patil, F. (1998a). Moving target classification and tracking from real-time video. In *WACV*.

Lipton, A., Fujiyoshi, H., and Patil, R. (1998b). Moving target classification and tracking from real-time video. In *IEEE WACV*.

Logothetis, N. and Sheinberg, N. (1996). Visual object recognition. *Annual Review of Neuroscience*, 19:577–621.

Lowe, D. G. (1992). Robust model-based motion tracking through the integration of search and estimation. *International Journal of Computer Vision*, 8(2):113–122.

M. Betke, E. H. and Davis, L. (1997). Highway scene analysis. In *IEEE Conference on Intelligent Transportation Systems*.

M. Betke, E. H. and Davis, L. (2000). Real-time multiple vehicle detection and tracking from a moving vehicle. *Machine Vision and Applications*, 12:69–83.

Michie, D., Spiegelhalter, D. J., and Taylor, C. C. (1994). *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, New York.

Neskovic, P. and Cooper, L. N. (2002). A recognition system that uses saccades to detect cars from real-time video streams. In *International Conference on Neural Information Processing*.

Neskovic, P. and Cooper, L. N. (2003). Providing context for edge extraction in application to detection of cars from video streams. In *International Conference on Engineering Applications of Neural Networks*.

Papageorgiou, C. and Poggio, T. (1999). A trainable object detection system: Car detection in static images. A.i. memo no. 1673, MIT.

Rajapakse, J. C. and (Eds.), L. W. (2004). *Neural Information Processing: Research and Development*, chapter Biologically inspired recognition system for car detection from real-time video streams, pages 320–334. Springer - Verlag.

Reilly, D. L., Cooper, L. N., and Elbaum, C. (1982). A neural model for category learning. *Biological Cybernetics*, 45:35–41.

S-W. Lee, H. H. B. and (Eds.), T. P. (2000). *Biologically motivated computer vision*. Berlin: Springer-Verlag.

Schneiderman, H. and Kanade, T. (2000). A statistical method for 3d object detection applied to faces and cars. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Scofield, C. L., Reilly, D. L., Elbaum, C., and Cooper, L. N. (1987). Pattern class degeneracy in an unrestricted storage density memory. In Anderson, D. Z., editor, *Neural Information Processing Systems*. American Institute of Physics.

Stone, C. J. (1977). Consistent nonparametric regression. *Annals of Statistics*, 5:595–645.

Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition*.

von der Malsburg, C. (1999). The what and why of binding: The modeller s perspective. *Neuron*, 24:95–104.

Wang, J., Neskovic, P., and Cooper, L. N. Neighborhood size selection in the k-nearest neighbor rule using statistical confidence. *submitted to Pattern Recognition*.

Wang, J., Neskovic, P., and Cooper, L. N. (2003). Partitioning a feature space using a locally defined confidence measure. In *Joint 13th International Conference on Artificial Neural Networks and 10th International Conference on Neural Information Processing*.

Wang, J., Neskovic, P., and Cooper, L. N. (2004). Context-based tracking of object features. In *International Joint Conference on Neural Networks*.

Webber, M., Welling, M., and Perona, P. (2000). Towards automatic discovery of object categories. In *Proc. CVPR*.

Wiskott, L., Fellous, J., Kruger, N., and v.d.Malsburg, C. (1997). Face recognition by elastic graph matching. *IEEE PAMI*, 7(19):775–779.

Z. Sun, G. B. and Miller, R. (2002). Quantized wavelet features and support vector machines for on-road vehicle detection. In *IEEE International Conference on Control, Automation, Robotics and Vision*.